Spotting Culprits in Epidemics: How many and Which ones?

B. Aditya Prakash
Computer Science Department
Virginia Tech.
Email: badityap@cs.vt.edu

Jilles Vrekeen
Computer Science Department
University of Antwerp
Email: jilles.vreeken@ua.ac.be

Christos Faloutsos
Computer Science Department
Carnegie Mellon University
Email: christos@cs.cmu.edu

***Abstract*—Given a snapshot of a large graph, in which an infection has been spreading for some time, can we identify those nodes from which the infection started to spread? In other words, can we reliably tell who the culprits are? In this paper we answer this question affirmatively, and give an efficient method called NetSleuth for the well-known Susceptible-Infected virus propagation model.**

Essentially, we are after that set of seed nodes that best explain the given snapshot. We propose to employ the Minimum Description Length principle to identify the best set of seed nodes and virus propagation ripple, as the one by which we can most succinctly describe the infected graph.

We give an highly efficient algorithm to identify likely sets of seed nodes given a snapshot. Then, given these seed nodes, we show we can optimize the virus propagation ripple in a principled way by maximizing likelihood. With all three combined, NetSleuth can automatically identify the correct number of seed nodes, as well as which nodes are the culprits.

Experimentation on our method shows high accuracy in the detection of seed nodes, in addition to the correct automatic identification of their number. Moreover, we show NetSleuth scales linearly in the number of nodes of the graph.

***Keywords*-culprits; epidemics; diffusion; seeds;**

I. Introduction

When considering large graphs, epidemics are everywhere. For social networks, infectious diseases like the flu are prime examples, but hypes/memes are similarly epidemic in nature; whether it is friends discussing that latest gadget or phone, or sharing a funny video, there are nodes 'infecting' each other. Similarly, a computer virus can cause an epidemic in a computer network, as can a contaminant in a water distribution network. In each of these cases, given a single snapshot of a partially infected network, an important and challenging research question is how we can reliably identify those nodes from which the epidemic started; whether for inoculation to prevent future epidemics, or for exploitation for viral marketing.

As such, given a snapshot of a large graph $G(\mathcal{V}, \mathcal{E})$ in which a subset of nodes $\mathcal{V}' \subseteq \mathcal{V}$ is currently infected, and assuming the Susceptible-Infected (SI) propagation model, we consider the problem of how to efficiently and reliably find those seed nodes $\mathcal{S} \subseteq \mathcal{V}'$ from which the epidemic started, without requiring the user to choose the number of seed nodes in advance. In other words, we address the questions: *How many culprits are there, and who are they?*

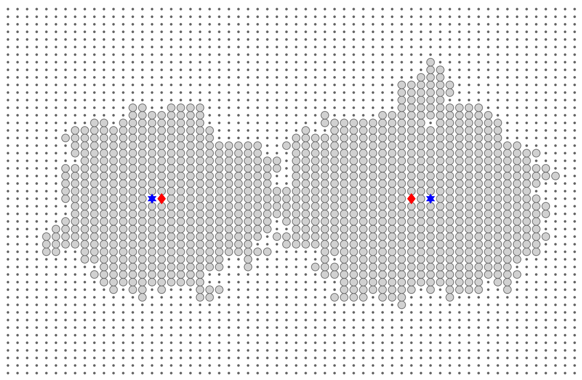


Figure 1. Example: Culprits, *how many*, and *which ones*? A snapshot of a 2D grid in which an infection has been stochastically spreading. Grey circles are infected nodes, while Grey dots are un-infected. The 2 Blue stars denote the true seeds. The 2 Red diamonds denote the seeds automatically discovered by NetSleuth—that is, both in *number* (two) and *location* (being spatially very close to the true seeds).

We propose to employ the Minimum Description Length (MDL) principle [1] to identify that set of seed nodes, and that virus propagation ripple starting from those nodes that best describes the given snapshot. We give an highly efficient algorithm to identify likely seed nodes, and show we can easily optimize the description length of the virus propagation ripple for a given seed set by greedily maximizing likelihood. As such, we can identify the best set of seed nodes in a principled manner, without having to choose k, the number of seed nodes in advance.

As an example, consider Figure 1. It depicts an example grid-structured graph, in which a subgraph has been infected by a stochastic process starting from two seed nodes. The plot shows the true seed nodes, as well as the seed nodes automatically identified by NetSleuth; it finds the correct number of seed nodes, and places these where a human would; in fact, the discovered seeds have a higher likelihood for generating this infected subgraph than the true seed nodes.

We develop a two step approach by first finding high-quality seeds given the number of seeds, and then using our carefully designed MDL score to pinpoint the true number of seeds. For the first part, we use the notion of 'exoneration' from the un-infected frontier—e.g., in Figure 1 the nodes on the edge of the infected snapshot are unlikely to be

Table I
COMPARISON BETWEEN THREE CULPRIT-IDENTIFYING METHODS: NETSLEUTH, RUMOR-CENTRALITY [2], AND EFFECTORS [3]

	infection model	$k>1$	automatically determines k	$O(\cdot)^\dagger$
NETSLEUTH (our method)	SI	✓	✓	Linear
Rumor-centrality [2]	SI	–	–	Quadratic
Effectors [3]	IC	✓	–	Quadratic

† Running time given for arbitrary graphs.

the culprits due to the large number of un-infected nodes surrounding them. Based on this idea, we develop a novel 'submatrix-laplacian' method to find out the best seed sets given a number of seeds (see Section IV for more details). Given these seed-sets, we also give an efficient algorithm to compute the MDL scores, thus finding the number of seeds in a parameter-free way.

Although network infection models have been researched extensively, identifying the seed nodes of an epidemic is surprisingly understudied. We are, however, not the first to research this problem. Recently, Shah and Zaman [2], [4] developed rumor-centrality for identifying the *single* source node of an epidemic. In contrast, we allow for *multiple* seed nodes, and automatically determine their number. Lappas et al. [3] studied the 'Effectors' problem of identifying k seed nodes in a steady-state network snapshot, under the Independent Cascade (IC) model. In contrast, we study the SI model, allow the snapshots from any time during the epidemic, and our approach is parameter-free as by MDL we can automatically identify the best value for k. Furthermore, and very importantly for large graphs, in comparison our method is computationally much more efficient. Table I gives a comparison of NETSLEUTH to these methods. We discuss related work in more detail in Section VI.

Experimentation shows that NETSLEUTH detects seed nodes and automatically identifies their number, both with high-accuracy. With synthetic data we show it can handle difficult fringe cases, and is in agreement with human intuition. We show we reliably identify the correct number of seed nodes on real data, and also that our detected seeds are of very high quality (measured by multiple metrics). Finally, we show our method scales linearly with the number of edges of the graph.

The rest of the paper is organized in the typical way: preliminaries, our problem formulation and method, experiments, related work and then conclusion.

II. PRELIMINARIES

In this section we give notation, and introduce MDL and the infection spreading model we use.

A. Notation

Table II gives some of the notation and symbols we will be using in the paper. We consider undirected, unweighted graphs $G = (\mathcal{V}, \mathcal{E})$ of $N = |\mathcal{V}|$ nodes. All logarithms in this paper are to base 2, and we adopt the standard convention that $0 \log 0 = 0$. We denote the transpose of any matrix or vector V as V^T. Finally note that L_A is a *submatrix* of $L(G)$, *not* the laplacian matrix of G_I.

Table II
TERMS AND SYMBOLS

Symbol	Definition and Description
SI model	Susceptible-Infected model
β	attack probability of the virus in the SI model
$G = (\mathcal{V}, \mathcal{E})$	graph under consideration
$G_I = (\mathcal{V}_I, \mathcal{E}_I)$	given infected subgraph of G
R	ripple, list of sets of nodes how virus propagates
N	$\lvert\mathcal{V}\rvert$, number of nodes in graph G
N_I	$\lvert\mathcal{V}_I\rvert$, number of nodes in graph G_I
$d(i)$	degree of node i
$\mathcal{F}$	set of un-infected nodes having at least one infected neighbor (in $\mathcal{V}_I$)
$\mathcal{E}_F$	set of edges connecting nodes in $\mathcal{F}$ to $\mathcal{V}_I$
$A(G)$	adjacency matrix of graph G (size $N \times N$)
A	adjacency matrix of G_I (size $N_I \times N_I$)
$D(G)$	diagonal degree matrix of graph G
$L(G)$	laplacian matrix of G i.e. $L(G) = D(G) - A(G)$
L_A	*submatrix* (size $N_I \times N_I$) of $L(G)$ corresponding to the infected graph G_I
Q_{MDL}	MDL-based culprits quality measure (see § V)
Q_{JD}	set-Jaccard-distance-based culprits quality measure (see § V)

B. The Susceptible-Infected Model

The most basic epidemic model is the so-called 'Susceptible-Infected' (SI) model [5]. Each object/node in the underlying graph is in one of two states - Susceptible (S) or Infected (I). Once infected, each node stays infected forever. Each infected node tries to infect each of its neighbors independently with probability β in each discrete time-step, which reflects the strength of the virus.

Note that here $1/\beta$ defines a natural time-scale (intuitively it is the expected number of time-steps for a successful attack over an edge). As an example, if we assume that the underlying network is a clique of N nodes, under continuous time, the model can be written as: $\frac{dI(t)}{dt} = \beta(N - I(t))I(t)$, where $I(t)$ is the number of infected nodes at time t—the solution is the logistic function and it is invariant to $\beta \times t$.

C. Minimum Description Length Principle

The Minimum Description Length principle (MDL) [1], is a practical version of Kolmogorov Complexity [6]. Both embrace the slogan *Induction by Compression*. For MDL, this can be roughly described as follows.

Given a set of models $\mathcal{M}$, the best model $M \in \mathcal{M}$ is the one that minimizes $\mathcal{L}(M) + \mathcal{L}(\mathcal{D} \mid M)$, in which $\mathcal{L}(M)$ is the length in bits of the description of M, and $\mathcal{L}(\mathcal{D} \mid M)$ is the length of the description of the data encoded with M.

This is called two-part MDL, or *crude* MDL—as opposed to *refined* MDL, where model and data are encoded together [1]. We use two-part MDL because we are specifically interested in the model: the seed nodes and ripple that give the best description. Further, although refined MDL has stronger theoretical foundations, it cannot be computed except for some special cases. Note that MDL requires the compression to be *lossless* in order to allow for fair comparison between different $M \in \mathcal{M}$.

To use MDL, we have to define what our models $\mathcal{M}$ are, how a $M \in \mathcal{M}$ describes the data at hand, and how we encode this all in bits. Note, that in MDL we are only concerned with code lengths, not actual code words.

III. Our Problem Formulation

Next we formulate our problem in terms of MDL. Our goal is to obtain the most succinct explanation of 'what happened'. To do so, we require two ingredients: the first is a formal objective—a cost function—which we discuss in this section. The second is then an algorithm to find good solutions, which we give in Section IV.

Our cost function will consist of two parts, 1) scoring the seed set (Model cost) and 2) scoring the successive infected nodes starting from the seed (Data cost).

We assume that both sender and receiver know the layout of $G = (\mathcal{V}, \mathcal{E})$, but not which nodes are in $G_I = (\mathcal{V}_I, \mathcal{E}_I)$. This makes, using the general formulation of MDL in Section II-C, G_I the data $\mathcal{D}$ we want to describe using our models $\mathcal{M}$. As such, informally, our goal is to identify those nodes, and an infection propagation ripple starting from those nodes, by which G_I can most easily be described.

A. Cost of the Model

As our models we consider *seed sets*. A seed set $\mathcal{S} \subseteq \mathcal{V}_I$ is a subset of $|\mathcal{S}|$ nodes of G_I from which the infection starts spreading—the 'patients zero', so to speak. We denote by $\mathcal{L}(\mathcal{S})$ the encoded length, in bits, of a seed set $\mathcal{S}$.

To describe a seed set $\mathcal{S}$, we first have to encode how many nodes $\mathcal{S}$ contains. This number, $|\mathcal{S}|$, is upper-bounded by the number of nodes in G. Hence, by using straight-forward block-encoding we can encode $|\mathcal{S}|$ in $\log N$ bits, by which we spend equally many bits to encode either a small or a large number. In general, however, we favor small seeds sets: simple explanations. The MDL optimal Universal code for integers [7] is therefore a better choice as it rewards smaller seed sets by requiring fewer bits to encode their size. With this encoding, $\mathcal{L}_{\mathbb{N}}$, the number of bits to encode an integer $n \geq 1$ is defined as $\mathcal{L}_{\mathbb{N}}(n) = \log^*(n) + \log(c_0)$, where $\log^*$ is defined as $\log^*(n) = \log(n) + \log\log(n) + \cdots$, where only the positive terms are included. To make $\mathcal{L}_{\mathbb{N}}$ a valid encoding, c_0 is chosen as $c_0 = \sum_{J \geq 1} 2^{-\mathcal{L}_{\mathbb{N}}(j)} \approx 2.865064$ such that the Kraft inequality is satisfied.

To identify which nodes in G are seed nodes, we use the very efficient class of data-to-model codes [8]. A data-to-model code is essentially an index into a canonically ordered enumeration of all possible data (values) given the model (the provided information). Here, we know $|\mathcal{S}|$ unique nodes have to be selected out of N, for which there are $\binom{N}{|\mathcal{S}|}$ possibilities. Assuming a canonical order, $\log \binom{N}{|\mathcal{S}|}$ gives us the length in bits of an index to the correct set of node ids.

Combining the above, we now have $\mathcal{L}(\mathcal{S})$ for the number of bits to identify a seed set $\mathcal{S} \subseteq \mathcal{V}_I$ as

$$\mathcal{L}(\mathcal{S}) = \mathcal{L}_{\mathbb{N}}(|\mathcal{S}|) + \log \binom{N}{|\mathcal{S}|} \tag{1}$$

B. Cost of the Data given the Model

Next, we need to describe the infected subgraph G_I given a seed set $\mathcal{S}$. We do this by encoding the infection *propagation ripple*, or the description of 'what happened'. Starting from the seed nodes, per time step we identify that set of nodes that gets infected at this time step, iterating until we have identified all the infected nodes.[1]

Propagation ripples: More formally, a propagation ripple R is a list of node ids per time-step t, which represents the order in which nodes of G_I became infected, starting from $\mathcal{S}$ at time $t = 0$. Let us write $\mathcal{V}_I^t(\mathcal{S}, R)$ to indicate the set of infected nodes at time t starting from seed set $\mathcal{S}$ and following ripple R, with $\mathcal{V}_I^0 = \mathcal{S}$. For readability, we do not write $\mathcal{S}$ and R wherever clear from context. As such, a valid propagation ripple R is a partitioning of node ids $\mathcal{V}_I \setminus \mathcal{S}$ of G_I, where every node in a part is required to have an edge from a node $j \in \mathcal{V}_I^{t-1}$.

Clearly, however, not every ripple from the seed set to the final infected subgraph is equally simple to describe. For instance, the more infected neighbors an uninfected node has, the more likely it is that it will get infected, as it is under constant attack—therefore, it should be more succinct to describe that this node gets infected than it would for a node under single attack.

Frontier sets: To encode a ripple R, at each time t we consider the collection of nodes currently under attack given the SI model (i.e. non-infected nodes with currently atleast one infected neighbor, or if $t = 0$, neighbors to a seed-node $\in \mathcal{S}$). We refer to this set as $\mathcal{F}^t$, for the frontier-set at time t. Define *attack degree* $a(n)$ of a non-infected node n as the number of infected neighbor nodes it has at the current iteration, i.e. $a(n) = |\{j \in \mathcal{V} \mid e_{jn} \in \mathcal{E} \wedge X_j(t)\}|$, in which $X_j(t)$ is an indicator function for whether node j is infected at time t.

We divide $\mathcal{F}^t$ into disjoint subsets $\mathcal{F}_i^t$ per attack degree i, that is, into sets of nodes having the same attack degree. As such, we have $\mathcal{F}^t = \mathcal{F}_1^t \cup \mathcal{F}_2^t \cup \ldots$, and correspondingly $f_1^t, f_2^t, \ldots$ for the sizes of these subsets (we will drop using the t superscript, when clear from context).

Starting from the seed set, for every time step t the receiver can easily construct the corresponding frontier set $\mathcal{F}^t$—which leaves us to transmit which of the nodes, if any, in the frontier

[1]When not interested in the actual ripple R, one could encode G_I by its overall probability starting from $\mathcal{S}$. Obtaining this probability, however, is very expensive, even by MCMC sampling. As we will see in Sections IV and V computing a good ripple is both cheap and gives good results.

set got infected in the current iteration. As, however, the infection probabilities per attack degree differ, we transmit this information per $\mathcal{F}_d^t$.

Probability of Infection: The SI model assumes an attack probability parameter β—so, the independent probability p_d of a node in $\mathcal{F}_d$ being infected is: $p_d = 1 - (1-\beta)^d$. Given p_d we can write down the probability distribution of a total of m_d nodes being infected for each subset $\mathcal{F}_d$. This is simply a Binomial with parameter p_d i.e.

$$p(m_d \mid f_d, d) = \binom{f_d}{k} p_d^{m_d} (1-p_d)^{f_d - m_d}$$

Hence, as such, a value for β determines p.

Encoding a Wave of Attack: Given p, a probability distribution for seeing m_d nodes out of f_d infected given an attack degree d, we need $-\log p(m_d \mid f_d, d)$ bits to optimally transmit the value of m_d. That is, we encode m_d using an optimal prefix code—for which we can calculate the optimal code lengths by Shannon entropy [9]. Then, once we know both f_d and m_d, we can use code words of resp. $-\log \frac{m_d}{f_d}$ and $-\log 1 - \frac{m_d}{f_d}$ bits long to transmit whether a node in $\mathcal{F}_d$ got infected or not. This gives us

$$\begin{aligned}\mathcal{L}(\mathcal{F}^t) \quad = \quad -\sum_{\mathcal{F}_d^t \in \mathcal{F}^t} \Bigg(\log p(m_d | f_d, d) + m_d \log \frac{m_d}{f_d} \\ + (f_d - m_d) \log 1 - \frac{m_d}{f_d} \Bigg) \end{aligned} \qquad (2)$$

for encoding the infectees in the frontier set at time t.

Then, for the recipient to know when to stop reading, we have to transmit how many time steps until we have reached G_I. The number of iterations T will be transmitted just like $|\mathcal{S}|$, using $\mathcal{L}_{\mathbb{N}}$. For the ripple R, starting from the frontier-set defined by the seed nodes $\mathcal{S}$, we iteratively transmit which nodes got infected at $t+1$—which in turn allows the recipient to construct $\mathcal{F}^{t+1}$. Note that, by $\mathcal{L}(\mathcal{F}^t)$ we assume ripple R to be in time scale of $1/\beta$. That is, for low β we consider a lower time resolution than for high β. This is because the SI model displays a natural invariance of time-scale (see Section II-B). So we have ripple R that gives the infections at every $1/\beta$ time-steps.

With the above, we have $\mathcal{L}(R \mid \mathcal{S})$ for the encoded length of a ripple R starting at a seed set $\mathcal{S}$ as

$$\mathcal{L}(R \mid \mathcal{S}) = \mathcal{L}_{\mathbb{N}}(T) + \sum_{t}^{T} \mathcal{L}(\mathcal{F}^t) \qquad (3)$$

C. The Problem

With $\mathcal{L}(\mathcal{S})$ and $\mathcal{L}(R \mid \mathcal{S})$, we have as the total description length $\mathcal{L}(G_I, \mathcal{S}, R)$ of an infected subgraph G_I of G following a valid infection propagation ripple R starting from a set of seed nodes $\mathcal{S}$ by

$$\mathcal{L}(G_I, \mathcal{S}, R) = \mathcal{L}(\mathcal{S}) + \mathcal{L}(R \mid \mathcal{S})$$

Note that as G is constant over all seed sets $\mathcal{S}$ and ripples R, we can safely ignore it in the computation of the total encoded size, for its encoded length would be constant term and hence not influence the selection of the best model.

By which we can now formally state our problem.

Minimal Infection Description Problem *Given a snapshot of a graph $G(\mathcal{V}, \mathcal{E})$ of N nodes, of which the subgraph $G_I(\mathcal{V}_I, \mathcal{E}_I)$ of N_I nodes are infected, and an infection probability β, by the Minimum Description Length principle we are after that seed set $\mathcal{S}$ and that valid propagation ripple R for which*

$$\mathcal{L}(G_I, \mathcal{S}, R)$$

is minimal for the Susceptible-Infected propagation strategy.

Clearly, this problem entails a large search space - both in the possible seed-subsets of $\mathcal{V}_I$ and the possible propagation ripples given any seed-set. In fact, as shown by Shah and Zaman [2], even the problem of just finding *one* MLE seed for a given infected snapshot in an arbitrary graph is very hard (#P-Complete, equivalent to counting the number of linear extensions of a poset). Further, they give provable algorithms for one seed, on d-regular trees only. We hence resort to heuristics to tackle the problem on general graphs.

IV. Proposed Method

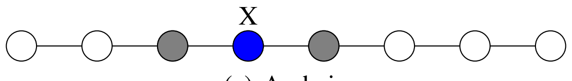
The outline of our approach is as follows: given a fixed number of seeds k, we identify a high-quality k-seed set. Given these seed nodes, we optimize the propagation ripple. With these two combined, we can use our MDL score to identify the best k.

A. Best seed-set given number of seeds—'Exoneration'

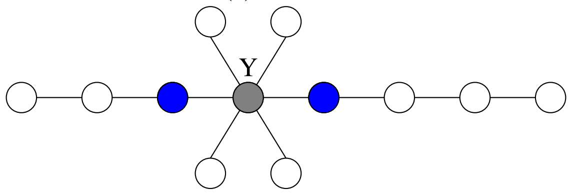
A central idea is that intuitively, un-infected nodes should provide some degree of 'exoneration' from 'blame' for the neighboring infected nodes. See Figure 2—it shows two illustrative examples of an infected chain (a) and a chain with a star in the middle (b) (colored nodes are infected and blue denote the true seeds). Note that while the node X is the most central among the infected nodes and is rightly the most likely seed, the node Y is not a likely seed because of the many un-infected nodes surrounding it. In fact, in this case the most likely starting points would be the two Blue nodes. Hence any method to identify the seed-sets should take into account the centrality of the infected nodes among the infected graph, but also penalize nodes for being too close to the un-infected frontier (the 'exoneration'). As we explain next, our method is able to do this in a principled manner.

B. Finding best single seed—Our Main Idea

We first explain how to find the best single seed and then how to extend it to multiple seeds. Jumping ahead, the main idea is as follows.



(a) A chain



(b) A chain-star

Figure 2. Centrality is not enough - effects of 'exoneration': Infection snapshot examples (colored nodes are infected, blue nodes are the true seeds) (a) Node X is the most central among the infected nodes; (b) Node Y is the most central among infected nodes, but the high count of non-infected neighbors 'exonerates' it.

Main Idea: The single best seed s^* is the one with the highest score in $\vec{u}_1$ i.e.

$$s^* = \arg\max_s \vec{u}_1(s)$$

where $\vec{u}_1$ is the *smallest* eigenvector of the laplacian submatrix L_A as defined in Table II. Next, we give the justification.

C. Finding the best single seed—Justification

From Section III, it is clear that nodes that are not in either the final frontier set $\mathcal{F}$ or $\mathcal{V}_I$ play no role, as they were not infectious nor could have been infected. Hence, WLOG, assume G contains only the infected subgraph G_I and the frontier set $\mathcal{F}$. Also, assume nodes are numbered in such a way that the first $|\mathcal{V} - \mathcal{V}_I|$ nodes are the un-infected nodes and the rest are the infected ones. If the total number of nodes in the graph is N, the number of infected nodes is N_I, then the number of un-infected nodes in G is $N - N_I$. Further notation is given in Table II.

Let $X_i(t)$ be the indicator (0/1) Random Variable denoting if node i in the graph is infected or not at time t (1 = infected, 0 = un-infected). Let $Y_{ij}(t)$ be the indicator random variable denoting if node j successfully attacks i at time t. Consider the following update equation for any node $i \in \mathcal{V}_I$:

$$\begin{aligned} X_i(t+1) &= X_i(t) + \\ &\quad (1 - X_i(t)) \times \\ &\quad \bigvee_{j\in\mathcal{N}(i)} Y_{ij}(t)(X_j(t) - X_i(t) + X_i(t)) \end{aligned} \tag{4}$$

Following the above equation, if $X_i(t) = 1$ then $X_i(t+1) = 1$, i.e., once a node is infected, it stays infected. Also if $X_i(t) = 0$, then $X_i(t+1) = \bigvee_{j\in\mathcal{N}(i)} Y_{ij}(t)X_j(t)$. Or in other words, an uninfected node may get infected only if an infected neighbor successfully transmits the infection. Additionally for any node $i \in \mathcal{V} - \mathcal{V}_I$, we define $X_i(t) = 0$, as these nodes were not infected at all during the infection process. Hence, the above equations exactly define a discrete-time SI process but with the constraint that the nodes in the given final frontier set *always stay un-infected*, thus enforcing the 'exoneration' discussed before. Hence we want to find the seed node which maximizes spread in this 'constrained' epidemic, which we show how to next.

For any node $i \in \mathcal{V}_I$, taking expectations both sides of Equation 4, and using the fact that for any indicator random variable X, $\mathbb{E}[X] = \Pr(X = 1)$, we get:

$$P_i(t+1) \quad = \quad P_i(t) + U - V \tag{5}$$

where,

$$\begin{aligned} P_i(t) &= \Pr(X_i(t) = 1) \\ U &= \mathbb{E}\left[\bigvee_{j\in\mathcal{N}(i)} Y_{ij}(t)(X_j(t) - X_i(t) + X_i(t))\right] \\ V &= \mathbb{E}\left[X_i(t) \times \bigvee_{j\in\mathcal{N}(i)} Y_{ij}(t)(X_j(t) - X_i(t) + X_i(t))\right] \end{aligned}$$

Clearly, as all the terms inside are positive,

$$V \geq 0, U \geq 0 \tag{6}$$

Also,

$$\begin{aligned} U &\leq \sum_{j\in\mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t) + P_i(t)) \\ &= \sum_{j\in\mathcal{N}(i)} A(G)_{ij}P_i(t) + \sum_{j\in\mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t)) \end{aligned}$$

as an infected node j attacks any of its neighbors i independently with probability $A(G)_{ij}$ (i.e. $\mathbb{E}[Y_{ij}(t)] = A(G)_{ij}$) and because by linearity of expectation, for any two events indicator random variables $\mathbb{1}_A$ and $\mathbb{1}_B$, we have $\mathbb{1}_A \vee \mathbb{1}_B = \mathbb{1}_A + \mathbb{1}_B - \mathbb{1}_A\mathbb{1}_B \Rightarrow \mathbb{E}[\mathbb{1}_A \vee \mathbb{1}_B] \leq \mathbb{E}[\mathbb{1}_A] + \mathbb{E}[\mathbb{1}_B]$. Also note that:

$$\sum_{j\in\mathcal{N}(i)} A(G)_{ij}P_i(t) \leq d_{max} \times P_i(t) \tag{7}$$

where d_{max} is the largest degree in graph G. Thus,

$$U \leq d_{max}P_i(t) + \sum_{j\in\mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t)) \tag{8}$$

From Equations 6 and 8, we can conclude that, for each node $i \in \mathcal{V}_I$:

$$\begin{aligned} P_i(t+1) &\leq P_i(t) + d_{max}P_i(t) \\ &\quad + \sum_{j\in\mathcal{N}(i)} A(G)_{ij}(P_j(t) - P_i(t)) \end{aligned}$$

Let $\sigma = 1 + d_{max}$. Recall that $\forall t,\ P_i(t) = 0$ for any *eventual* un-infected node $i \in \mathcal{V} - \mathcal{V}_I$. Let $\vec{P}(t) = [P_1(t), P_2(t), \ldots, P_N(t)]^T$ (over all the nodes in $\mathcal{V}$). Then we can write:

$$\vec{P}(t+1) \leq \sigma(I - \frac{1}{\sigma}M)\vec{P}(t) \tag{9}$$

where, the matrix M (size $N \times N$) is:

$$M = \begin{vmatrix} 0_{N-N_I,N-N_I} & 0_{N-N_I,N_I} \\ 0_{N_I,N-N_I} & L_A \end{vmatrix}$$

where we write $0_{N,M}$ for an all-zeros matrix of size $N \times M$. Let the subvector of $\vec{P}(t+1)$ corresponding to the infected nodes be written as $\vec{P}_I(t+1)$. Then continuing from above and using the upper bound as an approximation, we get:

$$\begin{aligned} \vec{P}_I(t+1) &\approx \sigma(I - \frac{1}{\sigma}L_A)\vec{P}_I(t) && (10) \\ &= \sigma(I - \frac{1}{\sigma}L_A)^t\vec{P}_I(0) && (11) \\ &= \sigma\sum_i \lambda_i^t \vec{u}_i\vec{u}_i^T\vec{P}_I(0) && (12) \end{aligned}$$

where, λ_i and $\vec{u}_i$ are the eigenvalues and eigenvectors of the matrix $I - \frac{1}{\sigma}L_A$. We have the following two lemmas:

Lemma 1. *The largest eigenvalue λ_1 and eigenvector $\vec{u}_1$ of the matrix $I - \frac{1}{\sigma}L_A$ are all positive and real.*

Proof: (Details omitted for brevity) The matrix $I - \frac{1}{\sigma}L_A$ is non-negative, and imagining $I - \frac{1}{\sigma}L_A$ as an adjacency matrix, the corresponding graph is irreducible, as graph G_I (adjacency matrix A) is connected. We then get the lemma due to the Perron-Frobenius theorem [10]. ■

Lemma 2. *The largest eigenvalue of matrix $I - \frac{1}{\sigma}L_A$ and the smallest eigenvalue of L_A are related as $\lambda_1(I - \frac{1}{\sigma}L_A) = 1 - \frac{1}{\sigma}\lambda_N(L_A)$.*

Proof: (Details omitted for brevity) It is easy to see that any eigenvalue $eig(I - \frac{1}{\sigma}L_A) = 1 - eig(\frac{1}{\sigma}L_A)$. As the matrices are symmetric, all the eigenvalues involved are real. By the Cauchy eigenvalue interlacing theorem [11] applied to $L(G)$, all the eigenvalues of any co-factor C_{LG} of $L(G)$ are positive. By the famous Kirchhoff's matrix theorem [12], the determinant of any co-factor C_{LG} is also non-zero as it counts the number of spanning trees of G. Also, it is well-known that the determinant of any matrix is just the product of its eigenvalues [11]. Hence, all eigenvalues of any co-factor matrix C_{LG} of $L(G)$ are strictly positive. We can similarly apply eigenvalue interlacing successively to a suitable C_{LG} and so on till we get to L_A (a principal submatrix of $L(G)$), and get that all eigenvalues of L_A are strictly positive. The lemma follows then. ■

Hence, the eigenvector $\vec{u}_1$ is also the eigenvector corresponding to the smallest eigenvalue of L_A.

Now, from Equation 12 and Lemma 1, we have:

$$\begin{aligned} \vec{P}_I(t+1) &= \sigma\lambda_1^t \sum_i \frac{\lambda_i^t}{\lambda_1^t}\vec{u}_i\vec{u}_i^T\vec{P}_I(0) && (13) \\ &\approx \sigma\lambda_1^t\vec{u}_1\vec{u}_1^T\vec{P}_I(0) && (14) \end{aligned}$$

assuming a substantial eigen-gap or 'big-enough' t. Now assuming that $\vec{P}_I(0)$ is all zero *except* for a single seed s for which it is 1, we can conclude that ultimately in our 'constrained' epidemic,

$$\begin{aligned} \forall i \in \mathcal{V}_I,\ Pr(X_i = 1|s) &\underset{\sim}{\propto} \vec{u}_1(i)\vec{u}_1(s) && (15) \\ \forall i \in \mathcal{V} - \mathcal{V}_I,\ Pr(X_i = 1|s) &= 0 && (16) \end{aligned}$$

Clearly the most likely single seed s^* would be:

$$s^* = \arg\max_s \left[\sum_{i\in\mathcal{V}_I} Pr(X_i = 1|s) + \sum_{i\in\mathcal{V}-\mathcal{V}_I}(1 - Pr(X_i = 1|s))\right]$$

Using Equations 15 and 16,

$$\begin{aligned} s^* &\approx \arg\max_s \vec{u}_1(s)\sum_{i\in\mathcal{V}_I}\vec{u}_1(i) \\ &= \arg\max_s \vec{u}_1(s) && (17) \end{aligned}$$

Hence, for a single seed, we just need to find the node with the largest score in $\vec{u}_1$ (which is also the smallest eigenvector of the laplacian submatrix L_A from Lemma 2).

D. Finding best k-seed set

Note that simply taking the top-k in the above eigenvector will not give good k-seed-sets due to lack of diversity. This is because the error in the upper-bound approximation used in Equation 11 will become larger due to increase in the norm of $\vec{P}_I(0)$. Hence, we treat the newly chosen seed, say s^*, as *un-infected*, effectively exonerating its neighbors and boosting diversity. We redo our computation on the resulting *smaller* infected graph, but a potentially larger frontier set—hence, we take the next best seed *given* the s^* that has already been chosen. So for any given k, we successively find the best next seed, given the previous choices, by removing the previously chosen seeds from the infected set and solving Equation 17. For example, in Figure 1, the top suspect (Red on the right) will have a lot of suspicious neighbors as well. Thus, using our exoneration technique, the algorithm will be forced away from them towards the remaining Red seed.

E. Finding a good ripple

As discussed before, once we find the best seed-set $\mathcal{S}_k$ for a given k, we optimize the propagation ripple of $\mathcal{S}_k$ to G_I to minimize the total encoded size. Recall from Section III that this involves minimizing $\mathcal{L}(R \mid \mathcal{S})$, which consists of two terms. First, we have the cost of encoding the length of the ripple, the number of time-steps. While $\mathcal{L}_{\mathbb{N}}$ does grow for higher values of T, in practice this term will be dwarfed by the actual encoding of the subsequent frontier sets. As such, minimizing $\mathcal{L}(R \mid \mathcal{S})$ essentially comes down to minimizing $-\sum_t^T \mathcal{L}(\mathcal{F}^t)$, or, in other words, maximizing the likelihood of the ripple R. Further recall that the SI model has a natural scaling invariance, $1/\beta$. As our score takes this into account, the ripple with the smallest description length should too.

Algorithm 1 NETSLEUTH

```
Input: $G(\mathcal{V},\mathcal{E}) \equiv G_I^* \cup \mathcal{F}^*$, $G_I^*(\mathcal{V}_I,\mathcal{E}_I)$ (the infected graph) and $\mathcal{F}^*$ (the frontier set).
Output: $\mathcal{S}$ = the set of seeds (culprits).
1:  $L(G) = D(G) - A(G)$, the Laplacian matrix corresponding to graph $G$.
2:  $\mathcal{S} = \{\}$
3:  $G_I = G_I^*$
4:  while $\mathcal{L}(G_I, \mathcal{S}, R)$ decreases do
5:      $L_A$ = the submatrix of $L(G)$ corresponding to $G_I$.
6:      $v$ = eigenvector of $L_A$ corresponding to the smallest eigenvalue.
7:      $next = \arg\max_i v(i)$
8:      $\mathcal{S} = \mathcal{S} \cup \{next\}$
9:      $R$ = ripple maximizing likelihood of $G_I$ from $\mathcal{S}$
10:     $G_I = G_I \backslash \{next\}$ (Graph $G_I$ with node $next$ removed)
11: end while
12: return $\mathcal{S}$
```

Hence, we design the following procedure. For each attack-degree set $\mathcal{F}_d$, at any iteration we scale the number of attacks by $1/\beta$ i.e. a set of size f_d is equivalent to a set of size f_d/β. Then, to get the overall MLE ripple, we adopt the following heuristic. We assume that the overall MLE ripple always performs a locally optimal next step. Hence this boils down to choosing the most-likely nodes to get infected at any given step, for a given frontier set $\mathcal{F}$.

It is well-known that a Binomial distribution $B(n,p)$ has its mode at $\lfloor (n+1)p \rfloor$. Using this fact, at any iteration t, taking into account the scaling, we can see that the most likely number of nodes infected in an attack-degree set $\mathcal{F}_d$ would be $n_d = \lfloor (f_d/\beta + 1) \times p_d \rfloor$—where p_d as defined before in Section III is the attack probability in the set $\mathcal{F}_d$. As such, we can simply uniformly choose this number of nodes from the $\mathcal{F}_d$, as each node in $\mathcal{F}_d$ is equally likely to be infected. We do this for every non-empty attack-degree set, for every iteration, until we have infected exactly the observed snapshot. This way, we obtain a most likely propagation ripple for any given seed-set $\mathcal{S}_k$ and can subsequently score it using MDL.

Finally, we stop getting more seeds when the MDL score for $\mathcal{S}_k$ increases as we increase k. Algorithm 1 gives the pseudo-code and Lemma 3 shows the running time for our algorithm NETSLEUTH.

Lemma 3 (Running Time of NETSLEUTH)**.** *The time complexity of* NETSLEUTH *is* $O(k^*(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I))$.

Proof: We keep finding $\mathcal{S}_k$ for each seed-set size until MDL tells us to stop. Hence the running time is $O(k^*(\mathcal{E}_I + T_{\text{RIPPLE}} + T_{\text{MDL}}))$, if k^* is the optimal seed-set size and T_{MDL} is the running time of computing the MDL score given the seed set size is k^*. Here we used the fact that calculating the eigenvector using the Lanczos method is approximately $O(E)$ (# edges) for sparse graphs.

The worst-case complexity T_{MDL} of calculating $\mathcal{L}(G_I, \mathcal{S}, R)$ for a given G_I, $\mathcal{S}$, and R, is $O(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I)$. The $\mathcal{L}(\mathcal{S})$ term is $O(1)$. For the $\mathcal{L}(R \mid \mathcal{S})$ term, we need to iterate over the ripple, which is at most $\mathcal{V}_I$ steps long. We only have to update the frontier set $\mathcal{F}$ when one or more nodes got infected, for which we then have to update the attack degrees of the nodes connected to the nodes infected at time t. Hence we traverse every edge in $\mathcal{E}_I + \mathcal{E}_F$, and every node in $\mathcal{V}_I$, which gives it the complexity of $O(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I)$.

Finally, the running time T_{RIPPLE} of computation of the MLE ripple for a given $\mathcal{S}_k$ is also $O(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I)$.

So the overall complexity of NETSLEUTH is $O(k^*(\mathcal{E}_I + \mathcal{E}_F + \mathcal{V}_I))$. ■

Hence NETSLEUTH is *linear* in the number of edges and vertices of the infected sub-graph and the frontier set, which makes our method scalable for large graphs (as compared to the methods in [2], [3] which, even for detecting a *single seed*, are $O(N^2)$).

V. EXPERIMENTS

Here we experimentally evaluate NETSLEUTH, in particular its effectiveness in finding culprits—whether it correctly identifies (a) *how many* as well as (b) *which ones*—and its (c) scalability.

A. Experimental Setup

We implemented NETSLEUTH in Matlab, and in addition we implemented the SI model as a discrete event simulation in C++. All reported results are averaged over 10 independent runs (so we generate 10 graphs for each seed set).

In our study we use both synthetic and real networks—we chose synthetic networks exemplifying different types of situations. We consider the following networks:

1) GRID is a 60×60 2D grid as shown in Figure 1.
2) CHAIN-STAR It is a graph of total 107 nodes. The first 7 nodes form a linear chain and the middle node has 100 additional neighbors.
3) AS-OREGON The Oregon AS router graph which is a network graph collected from the Oregon router views. It contains 15 420 links among 3 995 AS peers.[2]

For the experiments on AS-OREGON, we ran the experiments for true-seed count $k^* = 1, 2, 3$. So for each seed-set, we run a simulation till at least 30% of the graph is infected, and give the resulting footprint as input to NETSLEUTH. Note that, the larger the number of infections, the tougher it is to find the true seeds, as in the SI model any seed will eventually infect the whole graph with certainty. Finally, we make sure that the infected sub-graph was connected—otherwise, we just have separate problem instances.

[2]For more information see http://topology.eecs.umich.edu/data.html.

As discussed in the introduction and Section VI, the existing proposals for identifying culprits consider significantly different problems settings than we do (see Table I); the Rumor Centrality of Shah and Zaman [2], [4] can only discover *one* seed node, while Effectors of Lappas et al. [3] even consider a completely different infection model. As such we can not meaningfully compare performances, and hence here only consider NETSLEUTH.

Evaluation Function—a subtle issue: How to evaluate the goodness of a seed set? That is, in Figure 1, how close are the red seeds (recovered) from the blue seeds (true)? Notice that the recovered seeds may actually have *better* score than the actual ones, for the same reason that the sample mean of a group of 1D Gaussian instances gives lower sum-squared-distances than the theoretical mean of the distribution. Moreover, even for evaluation, it is intractable to compute the exact probability of observing the footprint from a given seed-set.

Thus we propose two quality measures. The first, Q_{MDL}, is based on our MDL: we report the ratio of the MDL score of our seeds, vs. the MDL score of the actual seeds i.e.

$$Q_{\mathrm{MDL}} = \frac{\mathcal{L}(G_I, \mathcal{S}, R)}{\mathcal{L}(G_I, \mathcal{S}^*, R^*)} \tag{18}$$

Clearly, the closer to 1, the better.

The second Q_{JD} intuitively measures the *overlap* of the footprint produced by a seed-set $\mathcal{S}$ and the input footprint $G_I(\mathcal{V}_I, \mathcal{E}_I)$. Clearly, the candidate seed-set $\mathcal{S}$ can produce n footprints, when we run n simulations; so we compute $\mathbb{E}[JD_{\mathcal{S}}(\mathcal{V}_I)]$, the average Jaccard distance[3] of all these n footprints, w.r.t. the true input footprint $\mathcal{V}_I$. As with Q_{MDL}, we normalize it with the corresponding score $\mathbb{E}[JD_{\mathcal{S}^*}(\mathcal{V}_I)]$ for the true seed-set, and thus report the ratio,

$$Q_{\mathrm{JD}} = \frac{\mathbb{E}[JD_{\mathcal{S}}(\mathcal{V}_I)]}{\mathbb{E}[JD_{\mathcal{S}^*}(\mathcal{V}_I)]} \tag{19}$$

Again, the closer to 1, the better.

B. *Effectiveness of* NETSLEUTH *in identifying How Many*

In short, NETSLEUTH was able to find the exact number of seeds for all the cases. Figures 3(a),(d),(g) show the MDL score as a function of $k = 1, 2, \ldots, 6$ seeds found by NETSLEUTH before stopping, for true seed-sets with (a) $k^* = 1$, (b) $k^* = 2$ and (c) $k^* = 3$ respectively on the AS-OREGON network. Note that the plots show near-convexity, with the minimum at the true k^*, justifying our choice of stopping after $j = 6$ iterations of increasing scores. It also shows the power of our approach, as we can easily recover the true number of seeds using a principled approach.

[3]We use the standard definition of Jaccard Distance between two sets $\mathcal{A}$ and $\mathcal{B} = 1 - \frac{|\mathcal{A} \cap \mathcal{B}|}{|\mathcal{A} \cup \mathcal{B}|}$.

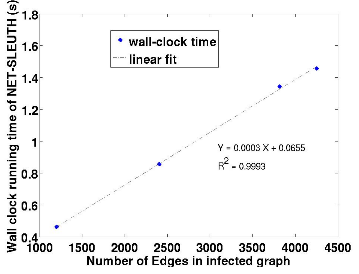


Figure 4. NETSLEUTH Scalability: Wall-clock running time (in seconds) for increasingly larger infected snapshots of AS-OREGON (as the complexity just depends on the size of the snapshot) $k^* = 1$. Each point average of 10 runs. Note that, as expected, the running time scales linearly.

C. *Effectiveness of* NETSLEUTH *in identifying Which Ones*

In short, in addition to finding the correct number of seeds, NETSLEUTH is able to identify good-quality seeds with high accuracy. For our synthetic graphs, NETSLEUTH is able to point out that there must have been exactly 2 seeds for both the GRID and CHAIN-STAR examples—respectively identified as the Red circles in Figure 1, and the Blue nodes in Figure 2(b)), agreeing with the ground-truth and intuition.

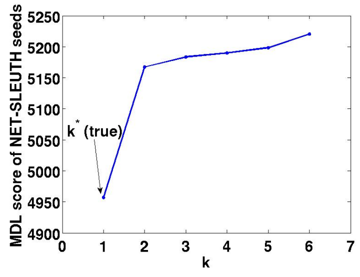
Figures 3(b-c),(e-f),(h-i) show the results of our experiments for different number $k^* = 1, 2, 3$ of true seeds on the AS-OREGON graph. We randomly selected 90 seed-sets of each size. We made sure that the seed-sets contained both well-connected and weakly connected nodes. Each point is an average of 10 runs.

Firstly, although not shown in the figures, NETSLEUTH was able to perfectly recover the true number of seeds in almost all cases. For each seed-set, we calculate $JD_{\mathcal{S}}(\mathcal{V}_I)$ for the seeds returned by NETSLEUTH and the true seeds and give the scatter plot in Figures 3(b)(e)(h) for true-seed count $k^* = 1, 2, 3$ respectively (rows). Hence points on or below the 45-degree line (solid blue) are better. Clearly, almost all points are concentrated near the diagonal, showing high quality. In fact, many points are exactly on the line, meaning we are able to recover the true seeds *perfectly* for many cases. We do not show similar plots with our MDL score due to lack of space.

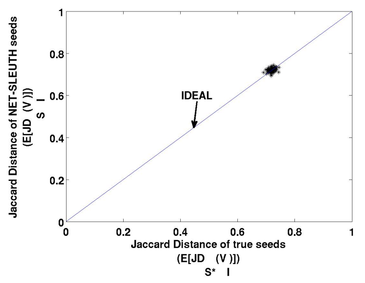
Next, we calculate Q_{MDL} and Q_{JD} averaged over all the different seed-sets (of the same size for $k^* = 1, 2, 3$). Results are shown in the bar plots (third column) of Figures 3(c)(f)(i). The true-seed scores are represented by the dotted line at 1, for both Q_{MDL} and Q_{JD}. Clearly, all of bars are close to 1, demonstrating that NETSLEUTH consistently finds very good culprits. Moreover, both Q_{MDL} and Q_{JD} quality metrics are similar in magnitude for all k^*'s — increasing our confidence in our results.

D. *Scalability*

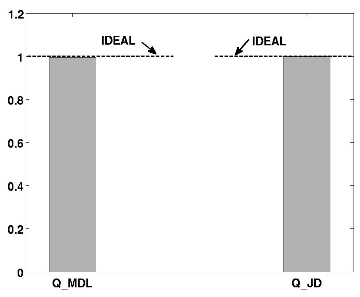
Figure 4 demonstrates the scalability of NETSLEUTH after running it on increasingly larger infected snapshots of AS-OREGON (as the complexity just depends on the size of the



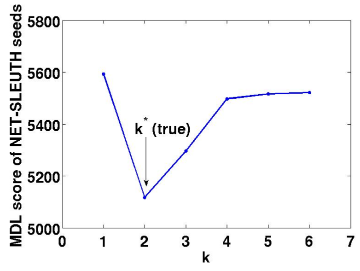
(a) $k^* = 1$



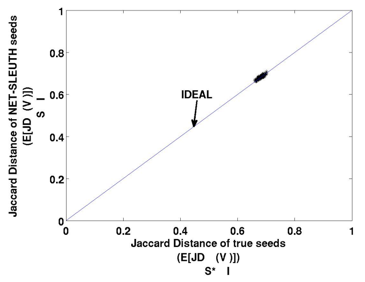
(b) Scatter plot of Jaccard scores



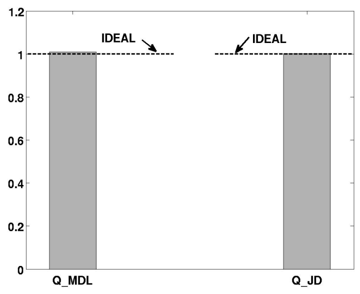
(c) Average Q_{MDL} and Q_{JD}



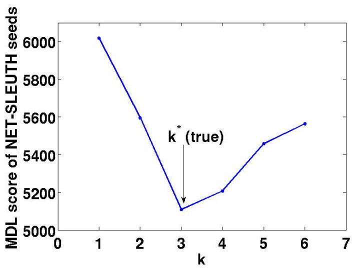
(d) $k^* = 2$



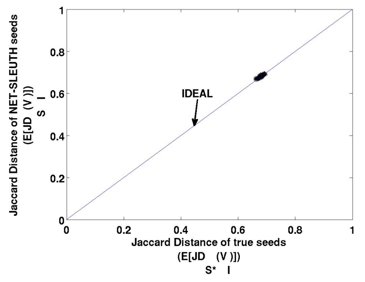
(e) Scatter plot of Jaccard scores



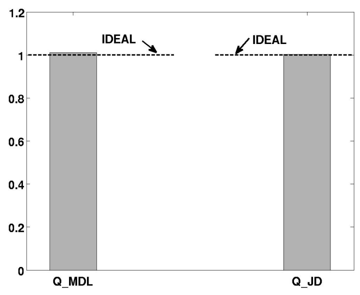
(f) Average Q_{MDL} and Q_{JD}



(g) $k^* = 3$



(h) Scatter plot of Jaccard scores



(i) Average Q_{MDL} and Q_{JD}

Figure 3. Effectiveness of NETSLEUTH in answering both *How many* and *Which ones* - Results of our experiments on the AS-OREGON graph for true-seed-count $k^* = 1, 2, 3$ (rows, subfigures (a-c), (d-f), (g-i) respectively). First column, (**a**),(**d**),(**g**), MDL scores as a function of k found by NETSLEUTH are near-convex; also we recover the true number in all cases. Second column, (**b**),(**e**),(**h**), scatter plots of Jaccard scores ($JD_x(\mathcal{V}_I)$) of NETSLEUTH seeds (y-axis) and the corresponding true seeds (x-axis). On or below the 45-degree line is better. Each point average of 10 runs. Note that for many runs the seeds identified by NETSLEUTH score exactly or even better than the true seeds. Third column, (**c**),(**f**),(**i**), average Q_{MDL} and Q_{JD} scores for the seeds returned by NETSLEUTH. Each bar represents the average over 90 different seed-sets. Note that all the bars are close to 1, indicating that we consistently find high-quality seed sets both with the Jaccard measure, and with the MDL measure.

snapshot). As expected from our Lemma 3, the running-time is linear on the number of edges of the infected graph.

VI. Related Work

As mentioned in the introduction, although diffusion processes have been widely studied, the problem of 'reverse engineering' the epidemic has not received much attention, except papers by Shah and Zaman [2], [4] and Lappas et al. [3]. Shah and Zaman [2], [4] formalized the notion of *rumor-centrality* for identifying the single source node of an epidemic under the SI model, and showed an optimal algorithm for *d-regular trees*. Lappas et al. [3] study the problem of identifying k seed nodes, or *effectors* of a partially activated network, which is assumed to be in *steady-state* under the IC (Independent-Cascade) model. In contrast, we allow for (a) multiple seed nodes, (b) a snapshot from any time during the infection, and (c) find the number of seeds automatically, even for general graphs. Finally we are also more efficient with linear time on edges of the infected graph. Also we are, to the best of our knowledge, the first to employ MDL with the goal of identifying culprits.

We categorize the rest of the related work into areas dealing with epidemic/cascade-style processes and problems related to them like epidemic thresholds, immunization and influence maximization. There is a lot of research interest in studying different types of information dissemination processes on large graphs in general, including (a) information cascades [13], [14], (b) blog propagation [15], [16], and (c) viral marketing and product penetration [17].

Epidemic Thresholds: The canonical text-book for epidemiological models like SI is Anderson and May [5]. Much research in virus propagation studied the so-called epidemic threshold, that is, to determine the condition under which an epidemic will not break out [18]–[22].

Influence Maximization: An important problem under the viral marketing setting is the influence maximization problem [23]–[26]. Another remotely related work is outbreak detection [27] in the sense that we aim to select a subset of '*important*' nodes on graphs.

Immunization: Another remotely related problem for such propagation processes is immunization - the problem of finding the best nodes for removal to stop an epidemic, with effective immunization strategies for static and dynamic graphs [28]–[30].

VII. Conclusions

In this paper we discussed finding culprits, the challenging problem of identifying the nodes from which an infection in a graph started to spread. We proposed to employ the Minimum Description Length principle for identifying that set of seed nodes from which the given snapshot can be described most succinctly. We introduced NetSleuth (based on a novel 'submatrix-laplacian' method), a highly efficient algorithm for both identifying the set of seed nodes that best describes the given situation, and *automatically* selecting the best number of seed nodes—in contrast to the state of the art.

Experiments showed NetSleuth attains high accuracy in detecting the seed nodes, as well as correctly identifying their number. Importantly, NetSleuth scales *linearly* with the number of edges of the infected graph, $O(\mathcal{E}_F + \mathcal{E}_I + \mathcal{V}_I)$, making it applicable on large graphs.

Future work can include extending our method to the popular SIR epidemic model.

Acknowledgments This material is based upon work supported by the Army Research Laboratory under Cooperative Agreement No. W911NF-09-2-0053, and the National Science Foundation under Grant No. IIS-1017415. J. Vreeken is supported by a Post-Doctoral Fellowship of the Research Foundation—Flanders (FWO).

References

[1] P. Grünwald, *The Minimum Description Length Principle*. MIT Press, 2007.
[2] D. Shah and T. Zaman, "Rumors in a network: Who's the culprit?" *IEEE TIT*, vol. 57, no. 8, pp. 5163–5181, 2011.
[3] T. Lappas, E. Terzi, D. Gunopulos, and H. Mannila, "Finding effectors in social networks," in *KDD*, 2010, pp. 1059–1068.
[4] D. Shah and T. Zaman, "Detecting sources of computer viruses in networks: theory and experiment," in *SIGMETRICS*, 2010, pp. 203–214.
[5] R. M. Anderson and R. M. May, *Infectious Diseases of Humans*. Oxford University Press, 1991.
[6] M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and its Applications*. Springer, 1993.
[7] J. Rissanen, "Modeling by shortest data description," *Annals Stat.*, vol. 11, no. 2, pp. 416–431, 1983.
[8] N. Vereshchagin and P. Vitanyi, "Kolmogorov's structure functions and model selection," *IEEE TIT*, vol. 50, no. 12, pp. 3265– 3290, 2004.
[9] T. M. Cover and J. A. Thomas, *Elements of Information Theory*. Wiley-Interscience New York, 2006.
[10] C. R. McCuler, "The many proofs and applications of Perron's theorem," *SIAM Review*, vol. 42, 2000.
[11] G. Strang, *Linear Algebra and its Applications*, 3rd ed. San Diego: Harcourt Brace Jonanovich, 1988.
[12] D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs: Theory and Applications, 3rd Ed.*, 1998.
[13] S. Bikhchandani, D. Hirshleifer, and I. Welch, "A theory of fads, fashion, custom, and cultural change in informational cascades," *Polit. Econ.*, vol. 100, no. 5, pp. 992–1026, 1992.
[14] J. Goldenberg, B. Libai, and E. Muller, "Talk of the network: A complex systems look at the underlying process of word-of-mouth," *Marketing Letters*, 2001.
[15] J. Leskovec, M. McGlohon, C. Faloutsos, N. Glance, and M. Hurst, "Cascading behavior in large blog graphs: Patterns and a model," in *SDM*, 2007.
[16] D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins, "Information diffusion through blogspace," in *WWW*, 2004.
[17] J. Leskovec, L. A. Adamic, and B. A. Huberman, "The dynamics of viral marketing," in *EC*, 2006.
[18] J. O. Kephart and S. R. White, "Measuring and modeling computer virus prevalence," in *SP*, 1993.
[19] R. Pastor-Santorras and A. Vespignani, "Epidemic spreading in scale-free networks," *Phys. Rev. Let. 86*, vol. 14, 2001.
[20] D. Chakrabarti, Y. Wang, C. Wang, J. Leskovec, and C. Faloutsos, "Epidemic thresholds in real networks," *TISSEC*, vol. 10(4), 2008.
[21] A. Ganesh, L. Massoulié, and D. Towsley, "The effect of network topology on the spread of epidemics," in *INFOCOM*, 2005.
[22] B. A. Prakash, D. Chakrabarti, M. Faloutsos, N. Valler, and C. Faloutsos, "Threshold conditions for arbitrary cascade models on arbitrary networks," in *ICDM*, 2011.
[23] M. Richardson and P. Domingos, "Mining knowledge-sharing sites for viral marketing," in *KDD*, 2002.
[24] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *KDD*, 2003.
[25] A. Goyal, W. Lu, and L. V. S. Lakshmanan, "Simpath: An efficient algorithm for influence maximization under the linear threshold model," *ICDM*, 2011.
[26] W. Chen, C. Wang, and Y. Wang, "Scalable influence maximization for prevalent viral marketing in large-scale social networks," *KDD*, 2010.
[27] J. Leskovec, A. Krause, C. Guestrin, C. Faloutsos, J. VanBriesen, and N. S. Glance, "Cost-effective outbreak detection in networks," in *KDD*, 2007, pp. 420–429.
[28] H. Tong, B. A. Prakash, C. E. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, and D. H. Chau, "On the vulnerability of large graphs," in *ICDM*, 2010.
[29] L. Briesemeister, P. Lincoln, and P. Porras, "Epidemic profiles and defense of scale-free networks," *WORM 2003*, 2003.
[30] B. A. Prakash, H. Tong, N. Valler, M. Faloutsos, and C. Faloutsos, "Virus propagation on time-varying networks: Theory and immunization algorithms," *ECML-PKDD*, 2010.